APPLICATION FOR UNITED STATES PATENT

by

MARK KIRKPATRICK,

SCOTT BASS,

DARIN MORROW

and

JOHN STROHMEYER

for

APPLICATION SERVER AND METHOD TO PERFORM HIERARCHICAL CONFIGURABLE DATA VALIDATION

SHAW PITTMAN LLP
1650 Tysons Boulevard
McLean, Virginia 22102-4859
703-770-7900

Attorney Docket No.: BS01-084

APPLICATION SERVER AND METHOD TO PERFORM HIERARCHICAL CONFIGURABLE DATA VALIDATION

FIELD OF THE INVENTION

[0001] The present invention relates to an application properties server and method to provide software validation service to clients. More particularly, the present invention relates to a system and a method for allowing applications software using established computer network protocols to execute hierarchical configurable data validation from a centralized database.

BACKGROUND OF THE INVENTION

[0002] Computer applications ("application servers") commonly require input data to be validated prior to additional processing. The validation requirements are often dynamic. For example, validation requirements for a telephone service provider's software applications may change based on changed customer service availability, newly available or no longer available customer services, the number of customer requests in a given period, the acceptable times for requests, the version of the software running, etc.

[0003] In today's networked environment, application servers run a variety of different software protocols (*e.g.*, J2EE Containers with CORBA orbs, J2EE Containers with RMI) and typically require a number of different data validations before performing other functions. As a result, a need exists for an application server that can dynamically maintain, process and efficiently run validations for a plurality of clients running different software protocols simultaneously.

[0004] Further, because validation needs often change, a need exists for a validation application server that can manipulate the validations run on specific fields of client validation requests without requiring extensive changes in software. Most computer software applications use configuration variables to alter its behavior without the need for regenerating code. This is most often done using text files. In today's Internet and networked environments this can cause administrative problems. This is largely because the same software application may be running on several machines, in several locations. Thus, in order to uniformly alter the behavior of all software applications, or clients, all files need to be accessible by the text files. This can cause great expense and significant administration problems. For one thing, security considerations often dictate that a text file employed to alter a software application must be on the same machine that is running the code. Therefore, the configuration file often must be replicated over several machines. If changes are made on one software application, they must also be made on all of the other applications. Errors can occur if the changes are not made consistently on all of the applications. Accordingly, a further need exists for an application server that will allow application server administrators to update the various validations done on fields of data without a new release of code.

SUMMARY OF THE INVENTION

[0005] The present invention is a system and method wherein application servers using standard software protocols may access a centralized, maintainable, validation application server coupled to a data schema for validation services on data. The client servers access the validation server via a number of methods including Internet applications, a Java RMI server, a CORBA gateway server and graphical screen interphase applications. The validation application server provides validation services on the data based on dynamically maintained, centrally stored, validation functions, and returns validation notifications to the client servers.

BRIEF DESCRIPTION OF THE DRAWINGS

[0006] Figure 1 is a schematic diagram of an overview of an embodiment of the present invention.

[0007] Figure 2 is a schematic diagram illustrating a specific embodiment of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

[0008] As shown in Figure 1, the present invention preferably includes an application properties server 100 for receiving validation requests from clients 400 and a storage mass 200 for storing centralized validation functions and data. As will be appreciated by those skilled in the art, validation properties application server 100 may be represented by one or more servers, even if located in different geographic locations. In the preferred embodiment of the present invention, depending on system resources, a number n of clients 400 may access the validation application server 100 for validation service via a number of methods including, for example, clients 401 using Internet applications, clients 402 using Java via a Java RMI server (not shown), clients 403 using CORBA via a CORBA gateway server (not shown), and graphical screen interphase applications.

[0009] Referring to Figure 1, according to an embodiment of the present invention, clients serve validation requests to the application server 100 which then accesses storage mass 200 using a hierarchical rule-based system 500 (Figure 2). The validation application server 100 identifies and accesses the stored data and performs validation services associated with the validation requests. Preferably tables of validation functions, or rules, implement the validation data, which may preferably be stored in a storage mass such as an Oracle database. As described hereinbelow, by utilizing a table-based storage system, the application server of the present invention can efficiently and dynamically perform validation services on validation requests provided by a number n of clients.

[0010] Referring to Figure 1, client 401 requests validated services related to data for, for example, long distance ordering information such as valid installation dates, available installation dates, or the allowable number of telephones. Storage mass 200 contains a plurality of data tables 210, 220, 230, . . . that will be described below. In response to client 401's validation requests, validation properties server 100 provides validation services to client 401 by accessing storage mass 200. Similarly, and preferably at the same time, a client 402 running a Java application program can use RMI via an RMI interface to interact with properties server 100 and validation of data based on information stored in storage mass 200. Finally, a third client, running a CORBA application 403 may request validation service on data related to, for example, Wireless Ordering. Again, validation properties server 100 accesses storage mass 200 and performs a validation service for client 403.

[0011] The validation data may be stored in a format such as Oracle or Lightweight Directory Access Protocol ("LDAP"). The information may be stored in another location or may be shared with other businesses. Preferably, validation data is stored in a table based system, more preferably in an Oracle database 200. As will be appreciated by those skilled in the art, use of an Oracle database allows for good performance, data integrity, and a wide variety of data management tools to administer the configuration data via an administration system 300. As will also be appreciated by those skilled in the art, Oracle database 200 may be represented by two or more databases located in separate geographical locations.

[0012] Figure 2 depicts a more specific embodiment and example of the present invention, wherein the database 200 consists of a table-based system of rules organized into three hierarchically-organized views: FIELD, CLASS and GLOBAL. The three views allow a hierarchical management of the validations to be performed on data fields received from the client 400 server. In the preferred embodiment, the three views are FIELD, CLASS and GLOBAL in order of precedence. Of course, the number of views may vary depending on the client's needs.

[0013] As will be appreciated, each of the FIELD, CLASS and GLOBAL views has an execution sequence. Utilizing an execution sequence, which provides a layered approach, and thereby a hierarchical approach to performing validation requests, yields efficient results. According to the execution sequence for a particular view, several validation methods can be orderly executed on data for a matching field.

[0014] Before providing a specific example, the FIELD, CLASS and GLOBAL views are explained. In the preferred embodiment, the FIELD view is the highest priority validation. Preferably, for efficiency reasons, the least amount of data is sorted by the FIELD view. If a FIELD name for the associated application is in this table that entry will dictate the validations to be performed.

[0015] As an example of one embodiment of the present invention, referring to Table 1, the FIELD view contains the following data:

Column Name	Description
Tag Name	Name of data field used to locate validations
Application Name	Application tag to differentiate field names from those in other applications.
Application Version	Application tag to differentiate field names from those in other versions of the same application.
Application User	Application tag to differentiate field names from those in other instances of an application and version for different users.
Execution Sequence	A number designating the order of execution for the 1 or more validation methods for an item meeting the previous criteria.
Validation Method	The name of an existing Java method to be called with the value of the field to be validated.
Validation Values ("PARM Data")	Used by the validation method to compare to the data value. Presence determined by validation method. Items are separated by a predefined character (generally a ",") (e.g., 1,5; 20020901, 20020601)
Comment	Description of desired rule. Used for documentation only.

TABLE 1

[0017] In the preferred embodiment, the CLASS view is the second-highest priority validation. The CLASS view is used if there is no matching entry in the FIELD view. In such a case, a lookup will be performed by the validation application server 100 on the passed field name. The validation application server 100 will check for class names that match the first part of the field name. An illustrative example is discussed below to describe the FIELD and CLASS view hierarchy.

[0018] Example 1: No Address_data. A client application server 400 accesses the validation application server 100 with the field name of Address_1. However, in reality, the user has input no Address_1 data. Thus, there is no Address_1 item in the FIELD view. There is, however, an entry in the class view for Address. Therefore the validation functions for the class Address will be performed on the data in Address_1.

[0019] As an example of one embodiment of the present invention, referring to Table 2, the CLASS view contains the following data:

[0020]

Column Name	Description
Tag Name	A generic string that will be used to match the field's name up to a defined character. (Date_1 will match up with Date)
Application Name	Application tag to differentiate field names from those in other applications.
Application Version	Application tag to differentiate field names from those in other versions of the same application.
Application User	Application tag to differentiate field names from those in other instances of an application and version for different users.
Execution Sequence	A number designating the order of execution for the 1 or more validation methods for an item meeting the previous criteria.
Validation Method	The name of an existing Java method to be called with the value of the field to be validated.
Validation Value 1 ("PARM Data")	Used by the validation method to compare to the data value. Presence determined by validation method. Items are separated by a predefined character (generally a ",")
Comment	Description of desired rule. Used for documentation only.

TABLE 2

[0021] Finally, in the preferred embodiment, the GLOBAL view is the most generic method of performing validation functions. Any field that does not have an entry in either the FIELD or CLASS view will be validated with the methods dictated for the associated application information. As this view is generic, preferably the most data is handled at the GLOBAL level, thereby improving efficiency. An example is discussed below describing the hierachry between the FIELD, CLASS and GLOBAL views.

[0022] Example 2: The field name is Residence_2. There is no Residence_2 item in the FIELD view. There is no Residence entry in the CLASS view. There is, however, a GLOBAL validation function called NotEmpty in the GLOBAL table for the application 400 (name, version and user) that requires data validation. Therefore the data for Residence_2 will be checked to see if it is not an empty field and validation properties server 100 will provide a positive return to client 400.

[0023] In the preferred embodiment, each of the FIELD, CLASS and GLOBAL views has an execution sequence for the associated validation functions that exist for that view. This provides a layered approach to validation. An example for describing the execution sequence is described below.

[0024] Example 3: Field Name: Date_1.

The FIELD table is as follows:

Tag Name	Application Name	Application Version	Application User	Execution Sequence	Validation Method	Validation Values
ClassOfService	Appl1	001	EMRS1	1	IsMember	0,1,2,3,4,5,6,7,8,9,0,A,F
DayOfWeek	Appl1	001	EMRS1	1	IsBetween	1,5

[0025] In this illustrative example, there is no match for Field Name: Date_1 in the FIELD view. Here, there are only validation executions in the FIELD view table for

ClassOfService and DayOfWeek. However, the validation properties server 100 recognizes that the CLASS view table has a matching item called Date. The CLASS view table has three entries (*i.e.*, three validation methods) to be performed. The CLASS view table is as follows:

Tag Name	**Application Name**	**Application Version**	**Application User**	**Execution Sequence**	**Validation Method**	**Validation Value 1**
Date	Appl1	001	EMRS1	10	IsBetween	01/01/2000,12/31/2002
Date	Appl1	001	EMRS1	1	NotEmpty	
Date	Appl1	001	EMRS1	5	IsValidDate	

[0026] In this exemplary example, based on the Execution Sequence of "1", the date is first checked for a non-empty field by validation method "NotEmpty." If it passes, based on the next rule, which has an execution sequence of "5", the date will be checked to see if it is a valid date format—using the IsValidDate method. If the date data passes that method, the date will be checked to make sure it is between January 1, 2000 and December 31, 2002 with the IsBetween method based on the Validation Value 1 PARM data in the CLASS view table. If the date data passes, then the data will be considered valid and the server 100 will return a positive return to the requesting client application server 400 signifying valid data. This example illustrates a "CLASS" level validation.

[0027] Figure 2 depicts an exemplary embodiment of the present invention. In this example, client application server Long Distance Delivery Ordering ("LD ORD") 400 seeks to validate data input by a customer for the weekday, the date available and the date of expiration related to a desired telephone service. Using a known software application protocol, application server 400 sends validation requests related to the data input by the customer to the validation application server 100. In this example, the weekday data is tag named "WkDay." Accordingly, the validation application server 100 generates an instruction to call the FIELD view table from the storage medium 200 for WkDay. In this example, the application server 100 automatically follows a priority of rules 500. Here, there is only one rule, *i.e.,* one Java Function, for WkDay. As shown in Figure 2, the Java method associated with the WkDay rule is an IsBetween method. Accordingly, the validation server 100 checks to see if the input data is between 1 and 5 based on the PARM data. If the data is between 1and 5, the validation server 100 returns a positive indication to client 400. If the input data is not between 1 and 5, the validation server 100 returns a negative indication to client 400.

[0028] Next, according to the example depicted in Figure 2, the validation server 100 performs validation service on validation requests for the data input for date available, which has been tagged DATE_AVL. Here, database 200 includes one rule in the FIELD view for data tagged DATE_AVL. Accordingly, validation server 100 performs a Java-based IsBetween function on the input data tagged DATE_AVL to check if the data is between June 1, 2002 and September 1, 2002. Here again, server 100 provides an appropriate response to client 400. In other embodiments, further services may be provided in response to a positive, or negative, result, such as additional data validations, or the generation of further functions.

[0029] Referring again to Figure 2, next, LD ORD 400 requests validation service on data related to the date of expiration for the requested telephone service. This data has been tagged DATE_EXP. The priority of rules 500 indicates that there are no rules in the FIELD view, *i.e.,* there are no validation functions, for DATE_EXP. Therefore, the validation application server 100 looks to the next level of rules: the CLASS view. Referring to the exemplary rules table 500, there are two rules in the CLASS view for

DATE: NotEmpty and IsValidDate. Accordingly, the application server 100 will read the execution sequences to determine which Java function to perform first. Java function NotEmpty has an execution sequence of "1" and Java function IsValidDate has an execution sequence of "5". Based on this sequence of execution set forth in the CLASS view, the validation server 100 executes first the NotEmpty function and then the IsValidDate function on the data. As will be appreciated by those skilled in the art, in this example, if there were no DATE rules, validation application server 100 would look for GLOBAL rules in the GLOBAL-view table. In the preferred embodiment, at least a NotEmpty validation function exists for the GLOBAL rules.

[0030] As will also be appreciated, by utilizing a centrally located storage system of dynamically maintainable validation rules, the present invention provides greater flexibility than known systems. For example, in the exemplary example discussed above, system administration 300 (Figure 1) can update the PARM data for the DATE_AVL rule and, accordingly, all applications 400 requesting validation services for data related to DATE_AVL are immediately updated.

[0031] In some instances, the number of validation requests may be large depending on the number n of application clients 400 using the server 100. Constant database 200 reads may cause delays in validation service. Therefore, in one exemplary embodiment, the validation server 100 will read the database 200 data 500 into memory on startup. Updates to the validation rules and values stored in the data tables can occur after system start up.

[0032] Two exemplary methods to handle dynamic table updates are described below. The first method is to restart the validation application server 100 each night during a maintenance window. This approach is a simple restart of the validation server. The application itself would not have to restart since it could detect the lost connection to the validation server and reconnect. This would be seamless to the applications and end user of the applications. Another exemplary method involves creating a refresh function in the validation server 100. Preferably the server 100 will use a REFRESH_HOURS parameter. The memory tables will be updated from the database 200 based on this parameter. Preferably, the REFRESH_HOURS will be greater than 8. As will be appreciated, keeping the data 500 in the application server 100 memory will improve validation performance and allow or maintaining the dynamic nature of the validation routines.

[0033] The most generic field type in Java is the string. In the preferred embodiment, all data passed to the validation server 100 will be treated as a string. This will allow applications 400 to change to more generic data without impact. This embodiment will provide an interface that is as generic as possible by establishing the interface as Strings (ASCII). An example of this concept is set forth below.

[0034] As an example, assume that a business requirement was originally for an integer value for the Class of Service variable. However, since the integer values can be type cast to String and passed to the validation server 100, modifications to the rules can be made quickly. Modifications to types would cause the validation server 100 to understand application knowledge and not data values and validation rules. Testing with the legacy system determines that the value of NumberOfTelephoneLines may also be a single alpha character in some legacy systems. According to the present invention, since the variable is stored as a string the client code is not affected. Accordingly, the validation functions in the data table can be changed to an IsMember method instead of an IsBetween method.

The validation value data will include all possible valid data values. This will change the validation from a check between two integer values (*e.g.*, a number between 1 and 5) to a check for a member of the a set (*e.g.*, 1,2,3,4,5,A,T,Y). This could be done without disturbing the running applications that utilize this validation service.

[0035] Proposed ValidatorClient CLASS methods include:

Method	Return Type	Arguments	Description
ValidatorClient	ValidatorClient	Application Name, Application Version, Application User	Class constructor. Also used to initialize Application data for subsequent calls
ValidatorClient	ValidatorClient		Class constructor.
~ValidatorClient	~ValidatorClient		Class destructor
set		Application Name, Application Version, Application User	Sets application data settings for the object.
isValid	Boolean	Field Name, Field Value	Sends data to the validator server for work. If the data passes the rules a TRUE is returned. Otherwise; a FALSE is returned.
IsValid	VRHashTable	FVHashTable	Sends data to the validator server for work. If the data passes the rules a TRUE is returned. Otherwise; a FALSE is returned.
RuleType	Integer	Field Name	Gets the rule type used for validation – 0 = None, 1= Field, 2= Class, 3=Global {Used primarily for development}

TABLE 3

[0036] In one embodiment, the client servers 400 can minimize network traffic using Field Value Hashtables. As will be appreciated, this will reduce the number of transactions to the validation application server 100 and improve performance. One call to the server 100 can contain an entire set of data in need of validation. The individual validation statuses will also be returned in a Hashtable. Preferably, the IsValid method is used to determine if all the data passed validation. If not, individual methods can be checked to determine problem areas.

[0037] Proposed FVHashTable CLASS methods include:

Method	Return Type	Arguments	Description
FVHashTable	FVHashTable		Constructor for a Field Value HashTable object.
AddToSet	Boolean	Field Name, Field Value	Add a field value pair to the FVHashTable. Return True on success.
RemoveFromSet	Boolean	Field Name	Remove a field value pair from the FVHashTable. Return True on success.
MemberValue	String	Field Name	Return the value of the specified key.

TABLE 4

[0038] Proposed VRHashTable CLASS methods:

Method	Return Type	Arguments	Description
VrashTable	VRHashTable		Constructor for a Value Return HashTable object.
AddToSet	Boolean	String Field Name, Boolean Valid	Add a field value pair to the VRHashTable. Return True on success.
RemoveFromSet	Boolean	String Field Name,	Remove a field value pair from the VRHashTable. Return True on success.
MemberValue	Boolean	Field Name	Return the validation status value of the specified key.
IsValid	Boloean		Returns a True if all values for the set are True. Otherwise; returns a False.

TABLE 5

[0039] As will be appreciated, according to the embodiments discussed above, two devices that are coupled can engage in direct communications, in indirect communications or a combination thereof. Embodiments of the present invention relate to data communications via one or more networks. The data communications can be carried by one or more communications channels of the one or more networks. Examples of a network include a Wide Area Network (WAN), a Local Area Network (LAN), the Internet, a wireless network, a wired network, a connection-oriented network, a packet network, an Internet Protocol (IP) network, or a combination thereof. A network can include wired communication links (e.g., coaxial cable, copper wires, optical fibers, and so on), wireless communication links (e.g., satellite communication links, terrestrial wireless communication links, wireless LANs, and so on), or a combination thereof.

[0040] In accordance with an embodiment of the present invention, instructions adapted to be executed by a processor to perform a method are stored on a computer-readable medium. The computer-readable medium can be a device that stores digital information. For example, a computer-readable medium includes a hard disk, a floppy disk, a tape and a compact disc read-only memory (CD-ROM), all as known in the art for storing software. The computer-readable medium is accessed by a processor suitable for executing instructions adapted to be executed. The term "adapted to be executed" is meant to encompass any instructions that are ready to be executed in their present form (*e.g.*, machine code) by a processor, or require further validation (*e.g.*, compilation, decryption, or provided with an access code, etc.) to be ready to be executed by a processor.

[0041] In describing representative embodiments of the present invention, the specification may have presented the method and/or process of the present invention as a particular sequence of steps. However, to the extent that the method or process does not rely on the particular order of steps set forth herein, the method or process should not be limited to the particular sequence of steps described. As one of ordinary skill in the art would appreciate, other sequences of steps may be possible. Therefore, the particular order of the steps set forth in the specification should not be construed as limitations on the claims. In addition, the claims directed to the method and/or process of the present invention should not be limited to the performance of their steps in the order written, unless that order is explicitly described as required by the description of the process in the specification. Otherwise, one skilled in the art can readily appreciate that the sequences may be varied and still remain within the spirit and scope of the present invention.

[0042] The foregoing disclosure of embodiments of the present invention has been presented for purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise forms disclosed. Many variations and modifications of the embodiments described herein will be obvious to one of ordinary skill in the art in light of the above disclosure. The scope of the invention is to be defined only by the claims appended hereto, and by their equivalents.